# Deep Learning Human Activity Recognition

David Browne, Michael.Giering, Steven.Prestwich

Insight-Centre of Data Analytics, University College Cork, Ireland
{david.browne,steven.prestwich}@insight-centre.org, gierinmj@utrc.utc.com

**Abstract.** Human activity recognition is an area of interest in various domains such as elderly and health care, smart-buildings and surveillance, with multiple approaches to solving the problem accurately and efficiently. For many years hand-crafted features were manually extracted from raw data signals, and activities were classified using support vector machines and hidden Markov models. To further improve on this method and to extract relevant features in an automated fashion, deep learning methods have been used. The most common of these methods are Long Short-Term Memory models (LSTM), which can take the sequential nature of the data into consideration and outperform existing techniques, but which have two main pitfalls; longer training times and loss of distant pass memory. A relevantly new type of network, the Temporal Convolutional Network (TCN), overcomes these pitfalls, as it takes significantly less time to train than LSTMs and also has a greater ability to capture more of the long term dependencies than LSTMs. When paired with a Convolutional Auto-Encoder (CAE) to remove noise and reduce the complexity of the problem, our results show that both models perform equally well, achieving state-of-the-art results, but when tested for robustness on temporal data the TCN outperforms the LSTM. The results also show, for industry applications, the TCN can accurately be used for fall detection or similar events within a smart building environment.

## 1 Introduction

In recent years, human activity recognition (HAR) and classification have gained momentum in both industry and academic research due to a vast number of applications associated with them. One area in particular where this research has huge interest is smart homes and the Internet of Things [14,19]. Other areas include crowd counting, health and elderly care, in particular fall detection, [20], [3]. Fall detection has been a popular area of research to enable more independent living for both the elderly and disabled within their own accommodation, but also within environments where cameras cannot be used due to data protection. There are two main approaches used for HAR: invasive and non-invasive. Invasive HAR involves wearing sensors to track humans to create a rich dataset for models to learn from, while non-invasive HAR allows humans to be monitored without any attached devices [21]. One way to do this is using WiFi signals, which are widely available in most buildings.

In HAR, the main activities in the classification task are sitting, standing, walking, lying down, falling down and human absence. All of these activities are of interest in the area of smart homes, while the falling down activity is of particular interest in health and elderly care, where cameras cannot be installed in private rooms but there is a need to monitor patients. This non-invasive, data sensitive method to alert staff to a patient falling is of great interest to the industry.

The idea behind HAR using WiFi signals is that the human body will affect the signal by reflection, and that different activities will show distinct characteristics. Initially most research in this area was carried out using the received signal strength (RSS), due to its accessibility [1]. With the development of the WiFi Network Interface Card, the data-rich Channel State Information (CSI) provides fine-grained information on multiple subcarriers [5]. The CSI carries the amplitude and phase for each subcarrier in orthogonal frequency-division multiplexing. By averaging the CSI across its subcarriers one can calculate the corresponding RSS, thus showing how much more information is carried on the CSI data compared to the RSS. The CSI data is typically converted into a spectrograph image, with the axes being time and channel, and the colour representing signal intensity. Since deep learning (DL) is state-of-the-art at image recognition, it is a very suitable choice for this application. Hence, we propose a DL method to classify human activities using CSI WiFI data.

There has been much recent work on HAR models, where feature engineering is required and the classification part of the tesk is preformed by traditional methods such as Support Vector Machines (SVM) or Decision Trees. Extracting features in this manner may result in fewer informative features and, because of the sequential nature of the data, subsets of features which are relevant to the classification task might be ignored. To overcome these issues, our method automatically learns the most discriminative features to distinguish each activity from each other. The main part of the proposed model is the TCN, consisting of a 1D fully-convolutional network and *causal convolutions*, which are convolutions where an output at time $t$ is convolved only with elements from time $t$ and earlier in the previous layer [2].

In this research, the data collection is performed in an indoor environment using a single access point. The data is transformed into a CSI image after some preprocessing, described below. We split the data into overlapping time windows which are then fed into our model. We explore the use of a CAE as a data reduction method, which has the added bonus of de-noising the data. The latent space information layer from the CAE is used as the input into a TCN. The TCN learns the sequential nature of the data, which is a vital property for accurate classification. We conducted extensive experiments, using both non-temporal and temporal variance, to validate our results. We carried out the same tests using a LSTM model, so that the results can be compared and contrasted to the more commonly-used sequential model.

The main contributions of this research are as follows. We introduce a novel approach for HAR using DL methods. The experiments are conducted on data

collected in an indoor environment which is setup to represent a home/office room, achieving state-of-the-art results, and also test the temporal robustness of our proposed method by testing on data collected over 7 days. Finally, to the best of our knowledge this is the first demonstration of the effectiveness of the TCN in this area, while previous DL approaches relied heavy on LSTMs.

This remainder of this paper is organised as follows. Section 2 discusses the related work on WiFi analytics and sequential models. We explain our proposed method in Section 3, and the collection of datasets in Section 4. The results are evaluated and discussed in Section 5, along with the experimental design. Finally, we conclude the paper in Section 6.

## 2 Related Work

Previously, HAR was performed using spatial features and SVMs to classify the feature representations, which could be either dense or sparse spatial points on histograms [4, 8, 19]. Later CNNs started to gain momentum, showing that DL could find relevant features, and hence outperform these methods quite significantly [7, 9, 23]. Ronao and Cho [16] argued that HAR has several real-world applications, in particular Ambient Assisted Living, due to the rising cost that an ageing population has on the economony. If more of the elderly could be given the opportunity to safely sustain themselves at home, then the pressure on the healthcare system would be reduced. They and others have shown how DL can take raw data signal from sensors: for example CNNs can be used to extract important features from accelerometer raw data [10, 12, 16].

Yousefi et al. [22] pre-processed channel state information CSI by means of principle component analysis to de-noise the signal, and then used a short-time Fourier transform to extract features. These features were then used as inputs to random forest (RF) and hidden Markov model (HMM) classifiers. The results were compared to a DL approach using a LSTM, which did not require de-noising or feature extraction as these are performed within the model. The RF and HMM models achieved 64.67% and 73.33% respectively. The LSTM model scored 90.5% accuracy: over 17% better than the HMM model and approximately 26% better than the RF. These results show that DL models can outdo classically methods, though the author noted that LSTMs were much slower to train. This is where our proposed model helps as it is significantly faster to train, as mentioned below.

Zou et al. [24] introduced a model called Auto-Encoder Long-term Recurrent Convolution Network (AE-LRCN), which was a DL approach to HAR. An autoencoder was used for representation learning and to remove inherent noise. The raw CSI time window was transformed into a latent space of 256 features, whereas our proposed method used a CAE not only to remove noise, but to compress the CSI time window into a latent space of 12 features. Next Zou et al. [24] used a convolutional network for feature extraction, which had 2 convolution layers followed by 2 fully connected layers. The proposed model does not require this step as the autoencoder had already performed aggressive feature selection. Finally, for sequential learning Zou et al. [24] implemented the pop-

ular LSTM, which has been shown to perform very well on this type of data. We introduce the TCN model, which is new to the area of HAR using CSI data, to learn the sequential nature of the data. We show that both the TCN and LSTM achieve state-of-the-art results in HAR, but the new proposed method is much more efficient. Wang et al. [18] introduced a DL-based channel selective activity recognition system called CSAR. This method requires considerable preprocessing, starting with channel quality evaluation and selection. They select the channels with an amplitude over a threshold, and neglect the others under the assumption that they are uninformative. Next they use channel hopping, where CSAR circularly hops through these selected channels, combining adjacent channels into an extended channel with higher bandwidth. Wang et al. [18] denoise the data by using a low-pass filter with a cut-off frequency of 100Hz, and PCA for data reduction and de-noising. Finally, the DL model implemented is the LSTM. In our proposed model the CAE denoises the signal, while the TCN — which is significantly faster than the LSTM — is used to learn the sequential nature of the CSI data. Wang et al. [18] used similar activities to our work, achieving on average over 95% accuracy, which compares well to our results.

Li et al. [11] used CSI data collected from multiple access points to classify four different activities. A DL model, consisting of multiple inputs into a convolutional neural network (CNN) combined into a fully connected layer then into a classifier, is compared to a SVM approach. The results show that the DL method learns to model the data better, achieving greater accuracy. Li et al. [11] transform the CSI data into spectrographs which are then divided into time windows. Qolomany et al. [15] compared a LSTM to one of the most common univariate models for time series, the Auto-Regressive Integrated Moving Average (ARIMA) model, using WiFi data. LSTM predictions were significantly better, reducing the root mean square error (RSME) by between 80.9% and 93.4%, showing once again that the DL approach is more promising in this area.

Trascau et al. [17] used sensor sequential data to detect actions of various types. The authors compared their model, which consisted of a TCN, to different setups of LSTMs. They noted that the TCN was faster to train than the LSTMs, and their results show that the TCN was about 5% more accurate. Another interesting result showed that the TCN was more robust on cross-subject and cross-view than LSTMs, which is also true in our work. Nair et al. [13] and Lea et al. [9] have successfully used TCNs for activity recognition. Both compared their TCN models to LSTMs, showing greater scores in the evaluation metrics they used. Nair et al. commented that the training time of the TCNs were in orders of magnitude quicker than that of LSTMs due to their parallelizable architecture. Their work also showed that TCNs were more robust and generalized better, possible due to their longer memory retention. Hou et al. [6] also used a TCN model on two hand gesture recognition datasets. The TCN was compared to CNNs, LSTMs and RNNs and outperformed them all. The results were also compared to approaches such as handcrafted features and histograms of gradients, but again showed a far superior ability to automatically extract

relevant features. Finally, Hou et al. stated that the TCN could be trained in an extremely short time, confirming Nair [13].

## 3 Deep Learning Activity Detection Chain

In this section we describe CAE-TCN (our new HAR technique), starting with the pre-processing of the CSI data, followed by the noise-removal and complexity reduction using the CAE, and finishing with the TCN model to learn the sequential latent-space representation of the activities, which will classify the probabilities as a particular activity.

Traditionally, approaches to this HAR problem was tackled using filters, such as the median or Butterworth filters to smoothing data, remove noise by discarding the first principle component, followed by a feature extraction phase usually involving some domain expert knowledge. The final stage consisted of a classifier, typically an SVM, to learn to map the selected features onto an n-dimensional space which could then be used to predict unseen samples.

We propose a novel DL approach that uses a CAE to find an embedding space for the time windows of the array created using the CSI data. The role of the CAE is twofold: (1) find this minimum embedding space, and (2) remove unwanted noise from the signal. The final part of the DL approach feeds this embedding space into a DL sequential model. Typically, for this stage an LSTM could be used, but we propose a different model: the recently-developed TCN. The TCN learns the inherent temporal dependencies needed to distinguish each of the activities that will be vital for the inference stage on new data. We will also show that although the TCN yields very similar results to the LSTM, the TCN is computationally more efficient.

### 3.1 Pre-processing

The CSI data is received in complex form, the real number representing the amplitude and the imaginary number representing the phase. As we are dealing with only a single transmitter and receiver setup, the phase is of little value and is discarded. An array of the amplitude of the 90 subcarriers, 30 within each frequency band, is saved to an array. This array consisted of 20 seconds of an activity, where the CSI data was sampled 5 times per second. This $90 \times 100$ array was then split into 4-second windows with a 50% overlap. Each 20-second activity array was transformed into a sequence of nine $90 \times 20$ time windows, which formed the dataset that the CAE was trained and validated on. The usual 80-10-10 training, validation and testing split was used to ensure no leakage into the training phases of the CAE or TCN.

### 3.2 Convolutional Auto-Encoder

The CAE is the first link in the proposed DL chain, and was fed with random samples, in each batch, of the 4-second time window arrays from the training

dataset. An input image is of size $90 \times 20$, representing the 90 subcarriers and 20 time steps which have been sampled at a rate of 5 per second. The CAE is a dimensional reduction model with an encoder and a decoder. The encoder reduces the size of the input while maintaining enough information to reconstruct the array. It maps the input array into a n-dimensional space where n is the latent embedding at the center of the CAE. The decoder then learns the remapping of this latent vector back into the input. The CAE is trained by trying to reduce the loss between the original array and the reconstructed array, while being forced through the bottleneck at the center of the CAE. The tighter the bottleneck is squeezed, the more efficiently the TCN will perform, but like all models it has to be guided to find the latent size: too much reduction and the model will fail to converge to a reasonable state. For these experiments, the encoder had 5 convolutional layers with 128 filters in the first 4 layers and 4 filters in the 5th (embedding) layer. The decoder was a mirror image of the first 4 layers. Each of the convolutional layers was followed by batch normalization, a leaky rectified linear unit, and dropout at a 25%, except for the embedding layer which had only batch normalization. A kernel size of $3 \times 3$ was selected throughout the network. The array was reduced from $90 \times 20$ to a size of $3 \times 1$, and since the embedding layer had 4 filters the latent layer had a vector size of 12.

A CAE is a unique DL model: the target output is the input but, as mentioned above, is squeezed through a narrow layer known as the embedding layer. This also helps to remove unwanted signal/noise in the array. Since the outputs $x'$ are trained to match the inputs $x$, the loss function used was the root mean squared error (RMSE). To avoid overfitting, an early stopping criterion was used to select the epoch in which the model was deemed to be trained. The optimizer used to train the CAE parameters was Stochastic Gradient Descent with a reducing learning rate. The initial learning rate was 0.1, and when no decrease in the loss within a patience value of 20 epochs occurred, it was reduced by a factor of 10. The minimum value of the learning rate was $10^{-5}$ and at this value, once the patience value was reached without improvement, training stopped. It should also be noted that the network was allowed to explore the loss space for 100 epochs before any decrease in the learning rate started. This gave an extra insurance that the network did not get stuck in a local minimum at the start of the training process. The sequences of nine 4 second $90 \times 20$ window arrays for an activity has been mapped onto a sequence of nine vectors of length 12, which was then be used to train and test the TCN.

### 3.3 Temporal Convolution Network

The default choice when dealing with sequence problems were LSTMs because of their powerful ability to find patterns in temporal dependencies in sequential data. A recent advancement from the CNN, known for image recognition achievements, is the TCN which uses a 1D fully-connected structure, in which each hidden layer is the same length as the input layer. To ensure that layers have the same size, zero-padding of length filter size minus one is added. Next, causal convolutions (defined above) are used to ensure no information leakage

from future to past. Dilated convolutions are used to reduce the model's look back history complexity which is size linear in network depth and filter size. Also, due to the TCN's growth in depth, convolutional layers are swapped for residual modules, which help to stabilize the network.

The TCN takes the sequential 12-vector embedding space as its input. The kernel size of the TCN is determined by the embedding size, which in this case is 12, and after experiments 100 nodes was found to be an optimal number. The number of layers in the TCN is directly related to the filter size and number of inputs, as each input is linearly connected to the output. The output of the model is the prediction of what activity was performed. The loss function used was softmax cross-entropy, which is a distance calculation between the probabilities from the softmax function and the one-hot-encodings of the activities. The same optimizer, reducing learning rate and early stopping criterion were used to train the TCN as above.

To compare with a LSTM, the TCN has lower memory requirements during training as the filters are shared across a layer, and back-propagation paths only depend on the depth of the network. LSTM captures only the temporal information of the data, whereas TCN captures both temporal and local information due to its convolutional operations. A big advantage TCN has over LSTMs is that when dealing with big data, TCNs can be parallelised because the convolutions can be run in parallel since the same filter is used in each layer, whereas in a LSTM the model has a looping process and runs sequentially, with time-step $t$ waiting until time-step $t$-$1$ has completed.

## 4 Datasets

This was a 6-class classification problem. The classes were sit, stand, walk, lay, fall and empty, which were chosen to represent the standard range a person would carry out on a daily basis. The lay class was based on lying down on a desk, used to simulate a person lying down on a bed. The fall class simulated a person falling and remaining on the ground. These two activities were purposely picked to show how this DL method could be used in a real-world situation.

Two sets of data were collected. The first, SetA, collected all the samples of each activity on the same day; the second, SetB, consisted of 3 samples of all the activities collected on a single day over a period of 7 different days. SetA had 20 samples over 180 seconds of each activity; the first and last 60 seconds was of the empty room, while the center 60 seconds was of the activity. SetB was collected differently, as 3 random sequences of each activity were performed for 60 seconds each, resulting in 1080 seconds of CSI data for each day. SetB had 126 samples in total, and was used to test the temporal robustness of the setup.

To ensure that only the relevant activity was performed in each sample, the center 40 seconds of the samples were subsectioned to be used, and were dissected into two 20-second samples. As explained above in the pre-process section, each of the 20-second samples were sequentially stacked into 50% overlapping 4 second

windows, which was used as the input data for the CAE. Both datasets are available from the corresponding author upon request.

## 5 Results

This section is divided into 4 subsections: an outline of the experimental design, followed by a subsection for each dataset, and finishes with a section on results for applicational purposes.

### 5.1 Experiment Design

All training and testing was conducted out on the NVIDIA GeForce GTX 1080 graphics card which has 8GB of memory. The OS used was Ubuntu 16.04.3, the Python version was 3.6, and the TensorFlow version was 1.9. A mini-batch size of 128 was used during training of the CAE, while because of the low number of training samples a batch size of 32 was used on the TCN and LSTM. For SetA 5-fold cross-validation was used with 70% of samples used for the training set, 10% used as the validation set and 20% used as the test set. For SetB 7-fold cross-validation was used, with 5 days of samples used for training, 1 for validation and 1 day for testing.

### 5.2 SetA

It can be seen from Table 1 that the TCN achieved state-of-the-art results, with an overall average accuracy of 99.2% accuracy. The only error it made was classifying a person sitting as a person standing. This result shows that this type of method could confidently be installed in a data protection restricted scenario, and perform at the highest level at HAR.

The LSTM did not achieve results as good as those of the TCN. It misclassified falling, sitting and standing, getting an overall average accuracy of 97.5%. It classed lying down as sitting down, which would be quite a reasonable mistake as both activities are performed close together. The next mistake was saying the room was empty when the person was standing. This could indicate that the person was standing very still, hence the change in the CSI data signal was not strong enough to detect the person standing. The final mistake the LSTM made was indicating that a person was standing, when they had actually fallen. This class would obviously be considered the worst misclassification in the context of an elderly care-home.

The final comparison between the two methods in this section was their computational speed and, as this method is designed for a smart-building setup, it is fair to use an ordinary CPU for this part of the experiment. Using 100,000 test samples on an i7 processor the TCN was 3.2 times faster than the LSTM model. This shows that the TCN model is not only much faster to train, but is also faster during inference.

| | Empty | Sit | Stand | Walk | Lay | Fall |
|---|---|---|---|---|---|---|
| Empty | 100 | 0 | 0 | 0 | 0 | 0 |
| Sit | 0 | 95 | 5 | 0 | 0 | 0 |
| Stand | 0 | 0 | 100 | 0 | 0 | 0 |
| Walk | 0 | 0 | 0 | 100 | 0 | 0 |
| Lay | 0 | 0 | 0 | 0 | 100 | 0 |
| Fall | 0 | 0 | 0 | 0 | 0 | 100 |

**Table 1.** CAE-TCN Confusion matrix.

| | Empty | Sit | Stand | Walk | Lay | Fall |
|---|---|---|---|---|---|---|
| Empty | 100 | 0 | 0 | 0 | 0 | 0 |
| Sit | 0 | 95 | 0 | 0 | 5 | 0 |
| Stand | 5 | 0 | 95 | 0 | 0 | 0 |
| Walk | 0 | 0 | 0 | 100 | 0 | 0 |
| Lay | 0 | 0 | 0 | 0 | 100 | 0 |
| Fall | 0 | 0 | 5 | 0 | 0 | 95 |

**Table 2.** CAE-LSTM Confusion matrix.

A small sample dataset was collected on a separate day, containing 3 samples of each activity, with the purpose of testing the proposed model for robustness against temporal variations. Unfortunately, the model performance was not acceptable, so various techniques were used to try to improve it. First, both the DL models parameters were altered using a grid-search setup. The number of layers and filters per layer in the CAE were increased and decreased. The kernel size was changed from $3 \times 3$ to $5 \times 5$, and the dropout rate and rate of regularization was varied to see if restricting or relaxing the network would help. As for the TCN and LSTM, the number of nodes, dropout rate and L2 regularization were explored, along with the kernel size for the TCN. None of these changes helped the CAE extract more useful features for temporal robustness, nor did the changes to the TCN/LSTM help to improve the classification accuracy by finding more informative patterns in the sequential data.

The next experiment tried to present the CAE with less noisy data as input. Initially, all the experiments transformed the raw data between 0-1 using the min-max formula. Related work suggests using PCA, and removing the first component did help as this component contained mostly noise. Using a low-pass filter was also suggested, and experiments were carried out using these together and individually, but with no improvement. Other pre-processing methods were explored, such as exponential smoothing, moving-average, de-trending and denoising, all commonly used with time series data. Background subtraction was also tested, using the blocks of data before the subject entered and after the subject had left the room. These blocks of data should, theoretically, represent the background or non-important information in the room. They were averaged across subcarriers, and subtracted from the activity data. They were also subtracted in blocks of data across time windows. Again, none of these experiments yielded any significant improvements.

This is why a second, more robust dataset, SetB, was collected. As seen in the next section, the accuracy results were slightly less impressive, though still within state-of-the-art range. However, the temporal robustness of the model was very good. Having a model that can predict so well over time has always been a challenging problem, and to show the proposed model can achieve these with only a few samples spread across time is valuable.

### 5.3 SetB

It can be seen from Table 3 that the TCN achieved state-of-the-art results with an overall average accuracy of 88.89% accuracy. It misclassed falling down for the empty room, an empty room for standing, standing for lying down, and lying down for falling down. Although the model had a few errors, it was still able to score nearly 90% classification accuracy on a 6-class HAR problem over temporal variance. This result could be improved with more temporal variance data, but the proposed model performed as expected under these conditions.

The LSTM did not achieve results as good as those of the TCN, only achieving an overall average accuracy of 80.56%. The error that can be considered the worst was that it misclassified falling down as an empty room more often than correctly classifying it. It was only able to correctly predict when the subject had fallen down approximately a third of the time, so this method could not confidently be installed in a real-world setting.

| | Empty | Sit | Stand | Walk | Lay | Fall |
|---|---|---|---|---|---|---|
| Empty | 83.3 | 0 | 0 | 0 | 0 | 16.7 |
| Sit | 0 | 100 | 0 | 0 | 0 | 0 |
| Stand | 16.7 | 0 | 83.3 | 0 | 0 | 0 |
| Walk | 0 | 0 | 0 | 100 | 0 | 0 |
| Lay | 0 | 0 | 16.7 | 0 | 83.3 | 0 |
| Fall | 0 | 0 | 0 | 0 | 16.7 | 83.3 |

**Table 3.** CAE-TCN Confusion matrix.

| | Empty | Sit | Stand | Walk | Lay | Fall |
|---|---|---|---|---|---|---|
| Empty | 100 | 0 | 0 | 0 | 0 | 0 |
| Sit | 0 | 100 | 0 | 0 | 0 | 0 |
| Stand | 0 | 0 | 83.3 | 0 | 16.7 | 0 |
| Walk | 0 | 0 | 0 | 100 | 0 | 0 |
| Lay | 0 | 0 | 33.3 | 0 | 66.7 | 0 |
| Fall | 50 | 0 | 0 | 0 | 16.7 | 33.3 |

**Table 4.** CAE-LSTM Confusion matrix.

### 5.4 Real-world application results on SetB

A new class is introduced into this section — the All class — which is the intersection set containing the activity being analysed and all the other activities. This class is therefore imbalanced at a ratio 5:1, and can be harder to classify correctly. No extra tuning was given to the model to handle the imbalance in the data, and the results are shown in Table 5 where the proposed model is trained to detect individual activities versus other states in the room. The first activity is empty, and it performs perfectly in all but one test, which is where the person has fallen, achieving 75% accuracy. This could be due to the person being out of sight between the transmitter and receiver. An increased number of access points or locations could solve this.

The next activity is lying, which against the empty room, sitting and walking achieved a perfect score, as shown in Table 5. When compared with the all class and to falling, it was able to classify them within 94.4% and 91.7% respectively, which are excellent results. When trying to differentiate between lying and standing, the proposed model was able to accurately predict over 83% of the time.

The last activity analysed is falling, shown in Table 5. This is probably the most important activity to detect, especially in elderly/ health care situations, as the longer a subject is on the floor the worse the potential consequences. The proposed model can easily identify falling down against sitting, standing and walking, and reasonably accurately against lying down at 91.7%. When distinguishing between falling down and all the remaining activities, a score of 88.9% is satisfactory due to the imbalance. The accuracy of 75%, where the model fails to accurately classify between an empty room and falling down, is explained above.

| | All | Empty | Sit | Stand | Walk | Lay | Fall |
|---|---|---|---|---|---|---|---|
| Empty | 100 | - | 100 | 100 | 100 | 100 | 75 |
| Lay | 94.4 | 100 | 100 | 83.3 | 100 | - | 91.7 |
| Fall | 88.9 | 75 | 100 | 100 | 100 | 91.7 | - |

**Table 5.** Results of the TCN method comparing binary states of HAR

## 6 Conclusion

In this paper, we presented a novel DL CAE-TCN, based on our experimental results, to accurately help solve the HAR problem. We design a Convolutional Auto-Encoder to helps to remove unwanted noise in the prepocessed CSI data, while compressing it through a bottleneck embedding layer. This compressed latent layer, vector size 12, was used to build a sequential TCN model for activity classification. By embedding the CSI window time steps onto a lower dimensionality, it greatly reduce the problem complexity, therefore allowing for real-world applicational purpose. The main contributions of this paper are that the CAE-TCN is computational more efficient, achieves state-of-the-art results, and is more robust than LSTMs on temporal variance in the CSI data.

In future work we plan to explore the use of transfer-learning to distill knowledge learnt in one room onto testing in a new environment. Other future work may include extending from single person to multi-person classification.

**Acknowledgement.** This work was supported in part by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289, and UTRC.

## References

1. Abdelnasser, H., Youssef, M., Harras, K.A.: Wigest: A ubiquitous wifi-based gesture recognition system. In: IEEE INFOCOM. pp. 1472–1480 (2015)
2. Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. CoRR abs/1803.01271 (2018)

3. Cippitelli, E., Fioranelli, F., Gambi, E., Spinsante, S.: Radar and rgb-depth sensors for fall detection: A review. IEEE Sensors Journal 17(12), 3585–3604 (2017)
4. Dollár, P., Rabaud, V., Cottrell, G., Belongie, S.: Behavior recognition via sparse spatio-temporal features. VS-PETS Beijing, China (2005)
5. Halperin, D., Hu, W., Sheth, A., Wetherall, D.: Tool release: Gathering 802.11 n traces with channel state information. ACM SIGCOMM Computer Communication Review 41(1), 53–53 (2011)
6. Hou, J., Wang, G., Chen, X., Xue, J.H., Zhu, R., Yang, H.: Spatial-temporal attention res-tcn for skeleton-based dynamic hand gesture recognition. In: Proc. of the ECCV. pp. 0–0 (2018)
7. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: Proc. of the IEEE conference on CVPR. pp. 1725–1732 (2014)
8. Laptev, I.: On space-time interest points. IJCV 64(2-3), 107–123 (2005)
9. Lea, C., Flynn, M.D., Vidal, R., Reiter, A., Hager, G.D.: Temporal convolutional networks for action segmentation and detection. In: Proc. of the IEEE Conference on CVPR. pp. 156–165 (2017)
10. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature 521(7553), 436 (2015)
11. Li, H., Ota, K., Dong, M., Guo, M.: Learning human activities through wi-fi channel state information with multiple access points. IEEE Com Mag 56(5) (2018)
12. Morales, J., Akopian, D.: Physical activity recognition by smartphones, a survey. Biocybernetics and Biomedical Engineering 37(3), 388–400 (2017)
13. Nair, N., Thomas, C., Jayagopi, D.B.: Human activity recognition using temporal convolutional network. In: Proc. of the 5th iWOAR. p. 17 (2018)
14. Pu, Q., Gupta, S., Gollakota, S., Patel, S.: Whole-home gesture recognition using wireless signals. In: Proc. of the 19th ACM MobiCom. pp. 27–38 (2013)
15. Qolomany, B., Al-Fuqaha, A., Benhaddou, D., Gupta, A.: Role of deep lstm neural networks and wi-fi networks in support of occupancy prediction in smart buildings. In: IEEE 19th Int. Conf. on HPCC; IEEE 15th Int. Conf. on SmartCity; IEEE 3rd Int. Conf. on DSS. pp. 50–57 (2017)
16. Ronao, C.A., Cho, S.B.: Human activity recognition with smartphone sensors using deep learning neural networks. Expert systems with apls 59, 235–244 (2016)
17. Trăscău, M., Nan, M., Florea, A.M.: Spatio-temporal features in action recognition using 3d skeletal joints. Sensors 19(2), 423 (2019)
18. Wang, F., Gong, W., Liu, J., Wu, K.: Channel selective activity recognition with wifi: A deep learning approach exploring wideband information. IEEE Transactions on Network Science and Engineering (2018)
19. Wang, G., Zou, Y., Zhou, Z., Wu, K., Ni, L.M.: We can hear you with wi-fi! IEEE Transactions on Mobile Computing 15(11), 2907–2920 (2016)
20. Wang, Y., Wu, K., Ni, L.M.: Wifall: Device-free fall detection by wireless networks. IEEE Transactions on Mobile Computing 16(2), 581–594 (2016)
21. Yatani, K., Truong, K.N.: Bodyscope: a wearable acoustic sensor for activity recognition. In: Proc. of ACM Conference on Ubiquitous Computing. pp. 341–350 (2012)
22. Yousefi, S., Narui, H., Dayal, S., Ermon, S., Valaee, S.: A survey on behavior recognition using wifi channel state information. IEEE Comms Mag. 55(10) (2017)
23. Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: Deep networks for video classification. In: Proc. of the IEEE conference on CVPR. pp. 4694–4702 (2015)
24. Zou, H., Zhou, Y., Yang, J., Spanos, C.J.: Towards occupant activity driven smart buildings via wifi-enabled iot devices and deep learning. Energy and Buildings 177, 12–22 (2018)